Chapter 1

Introduction

The exponential advancement of Information and Communication Technology (ICT) has drastically changed the way of living and thinking of people living in developed regions. But unfortunately people of developing countries are being left far behind due to lack of access to information and resources. This gap between the two domains, commonly known as Digital Divide, is ever increasing due to poverty, illiteracy, poor leadership and management etc. in the developing regions. We believe, education is the best way to fight the problem, since it deals with the major causes of this divide most closely. But the existing educational system and teaching-learning method directly blocks the possibilities. Since it teaches students to find out the ready made solution of the ready made problems, but does not foster them to explore their creativity and innovations.

Complete transformation of existing educational system could be the solution, but it is only hypothetical to think of the transformation instantly. Therefore we needed to focus on the most effective part of education, i.e. child education. This gave birth to Mero Sanu Sathi (MSS).

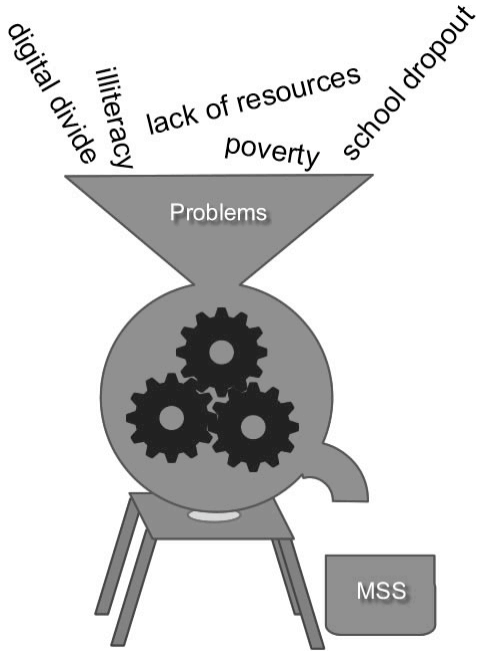


Fig1.1MSS mill model

MSS is a multidimensional tool for children to think with and explore the knowledge-base, helping them "Learn Learning". It consists of the harmony between "Child-Centric" Hardware and Software, based on the Constructionist Theory of Learning. As then name also suggests, MSS is a not a conventional teacher who imposes a rigid curriculum to children but a friend who plays with them and facilitates them in learning.

As we are following "Agile Development Model" for our project, MSS has passed through several adaptations. Currently we are developing "Activities" focused for Nepalese Children, which can be easily internationalized and localized to other locales. The six activities we have developed are Barnamala, Dunot, Chitrapati, Geet, Khel and Katha.

We have developed our activities to be fully compatible with the "Sugar Activities" of One Laptop Per Child (OLPC) laptops, which are being designed and developed as educational tools for the children of developing countries like Nepal. We have tested some of the activities in OLPC's A-Test Board successfully. Our target platform is OLPC laptop. But we do not mean that, the activities can only be run in those laptops. The activities can be run in almost all of the Desktop platforms being used today, as every bit of code is written in multi-platform Python Programming Language.

And it's a great pleasure for us to share all of our works in MSS freely to the developer community and the whole public as Open Content and Free and Open Source Software (FOSS/OC).

Problem Statement

Rapid development of the gap between digital haves (those people who have easy access to ICT) and haves not, has become a great threat for the developing countries like Nepal. This is due to the fact that the gap is being increased second by second. A bitter truth is that, these countries are not in the position to do anything even if they are aware of it; since they think that they have other high priority problems (e.g. poverty, illiteracy, unemployment etc.) to solve before investing to ICT. What most of the policy makers think is that ICT is a different field and an advanced thing to be dealt with everyday life.
But we, ICT students, need to prove that ICT is not only an advanced thing but also a tool that can be embedded in any field for better efficiency and effectiveness. But again the real implementation is far more different that just speaking the words. Embedding ICT into every field takes time and effort. But we believe that education is the best possible point to start with, as education shapes the man, man shapes the society and society shapes the country and it's entire development.

Transformation of education for transformation of the society is the objective of our project. To achieve the objective we have started with the foundation, i.e. children education. We have tried to use the techniques and technologies we've learnt to make the learning process most efficient and effective.

1.1 Rationale

MSS is a tool to be used for efficient and effective education, effective education is the tool for the development of the innovative and creative individuals and these individuals are the tools for the transformation of the whole society.

Here, efficient and effective education means the education, which does not tame the students to do a monotonous task, but assist/facilitate them to explore their innovation and creativity. Which is the basis of constructionist theory of learning. As Seymour Papert's principle says:

"Some of the most crucial steps in mental growth are based not simply on acquiring new skills, but on acquiring new administrative way to use what one already knows."

The development of a child passes through different stages, which should be taken in mind while making materials for them. The stages as defined by Jean Piaget are:

- Sensory Motor Stage (From birth to 2 years)—Experiencing through movement and senses and learn object Permanence)
- Preoperational Stage (Age 2 to 7)—Acquisition of motor skills
- Concrete Operational stage (Age 7 to 11)—Children begin to think Logically about concrete events
- Formal Operational Stage (After age 11)—Development of abstract reasoning

Whatever the context is there is an existence of commonality. This existence creates certain things to be learned. The Constructivism categorizes this learning in two main parts. First part is learning from assimilation, in which new things are learnt by manipulating the existing knowledge frame. Next part is adaptation, which requires accommodation and restructuring of existing frame of knowledge.

The new administrative way is usages of new tools, new tools to perceive the same knowledge base in better and efficient manner. At the dawn of century, technology has major role in our daily deeds the persistent shadow cast by technology in human activity at increasing day-by-day order. We cannot isolate ourselves from the effect stated. Either we favor or negate the aspect but we surely are attached to the fact. We people studying the applied technical field have a important driving role in bridging the gap between two bank of river categorized by using technology and starving technology which happens to exist at the same time in this little world we call home.

The technology should be embedded in our life but not by paralyzing us, they should assist us but not control us and more important they should know us rather than executing random segments of codes out of blues. The better place to embed fast growing computer technology as starting point can only be in educating, and that should start right where the learning starts, The technology should grow with learner and morph itself to meet the need presenting him only what he want and can handle. Our existing educational methodology and technology implement both are not in accordance with these facts. There needs to be synchronization between development, human, education and technology to familiarize technology where it is most important.

1.2 Requirements

MSS has been designed and developed as a friend and facilitator for children. The whole design is mainly focused on using the most effective modules available for child centric and constructionist learning. During the design process we kept the following requirements in mind:

1.3.1 User Requirements

MSS users are children and children. The system they interact should be in coherence with their thinking domains and the stage of their development. Our design motto was that, the system should not be simple but simplified. Everything should be activity based and child centric so that they do not feel the system as a teacher but as a friend and a facilitator.

Instead of ultra sophisticated user interfaces and presentation of vast facts and figures MSS requires seamless interface easily understandable and simple to handle. Learning should be a part of entertainment but not the burden for children. They should be provided enough options to explore the things themselves so that they do not feel learning as monotonous and boring.

1.3.2 Facilitator Requirements

With the introduction of MSS, there will be no teachers as in traditional teaching-learning process, but they will help children for learning-learning. The role of teacher is as a mere guide while using MSS. Hence from facilitator point of view interface should provide simple easily readable and explainable messages. And should be easy to guide users with as simple effort as possible.

1.3.3 System Requirements

Hence User and facilitator requires MSS to be simpler on their perspective, all burden are left to system. System must handle all errors presenting almost error free interface. Error messages should be as simple as possible if inevitable. All the calculation and Number crunching should be done by effective memory and process optimization. The activities should be space savvy, easily sharable and group oriented rather than individual.

1.3 History

MSS is following the agile model of project development. So it has passed through several adaptations and modifications from its birth. But our clear objective from start is: transformation of education for transformation of the society to make the world better place to live.

One of the motivators for us to start this project was “Combage”, a communication tool especially designed for rural and illiterate peoples. At first we started with embedded

system design ourselves. We went on finding out different hardware and software possibilities. We started to study about embedded ARM based systems among which StrongARM and Xscale were dragging our attention with embedded Linux distributions like Familier Linux, uCLinux, RTLinux etc. We also did not ruled out the possibility of hacking a bit expensive existing handheld systems like JuiceBox. At one point we were thinking of making the Java based applications with J2ME, which could be easily ported in systems with JRE.

At last we got the news of "$100 laptop" project (currently OLPC). And we found that OLPC has the similar objective to ours. Then we decided OLPC laptops as our hardware platforms. We started following the mainstream development of the OLPC Software with an A-Test Board and QEMU Emulator.

At the same time we did have the challenge to explore and learn about education and learning, which was entirely new stuff for engineering students like us. We started consulting educationalists, curriculum developers, teachers along with the extensive study of education and learning principles. We visited schools, interacted with our target groups and attended trainings.

Chapter 2

Literature Review

2.1 Agile Modeling

Agile modeling is a practice based adaptive methodology for effective modeling and documentation of software based systems. Simply put, agile modeling is a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and light weight manner. Agile model are more effective than other traditional models. The main values of agile modeling according to Ambler are:

- model with a purpose
- use multiple models
- travel light
- content is more important than representation
- know the model and tools you use to create them
- adapt locally.

2.2 Constructivism

Constructivism views learning as a process in which the learner actively constructs or builds new ideas or concepts based upon current and past knowledge. In other words, "learning involves constructing one's own knowledge from one's own experiences.Constructivist learning, therefore, is a very personal endeavor, whereby internalized concepts, rules, and general principles may consequently be applied in a practical real-world context.The teacher acts as a facilitator who encourages students to discover principles for themselves and to construct knowledge by working to solve realistic problems. This is also known as knowledge construction as a social process. We can work to clarify and organize their ideas so we can voice them to others. It gives us opportunities to elaborate on what they learned. We are exposed to the views of others. It enables us to discover flaws and inconsistencies by learning we can get good results. Constructivism itself has many variations, such as Generative Learning, Discovery Learning, and knowledge building. Regardless of the variety, constructivism promotes a student's free exploration within a given framework or structure.

Formalization of the theory of constructivism is generally attributed to Jean Piaget, who articulated mechanisms by which knowledge is internalized by learners. He suggested that through processes of accommodation and assimilation, individuals construct new knowledge from their experiences. Assimilation occurs when individuals' experiences are aligned with their internal representation of the world. They assimilate the new experience into an already existing framework. Accommodation is the process of reframing one's mental representation of the external world to fit new experiences. Accommodation can be understood as the mechanism by which failure leads to learning. When we act on the expectation that the world operates in one way and it violates our expectations, we often fail. By accommodating this new experience and reframing our model of the way the world works, we learn from the experience of failure.

It is important to note that constructivism itself does not suggest one particular pedagogy. In fact, constructivism describes how learning happens, regardless of whether the learner is leveraging their experiences to understand a lecture or attempting to design a model airplane. In both cases, the theory of constructivism suggests that learners construct knowledge. Constructivism as a description of human cognition is often confused with pedagogic approaches that promote learning by doing.

2.3 Python

Python is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code.

Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones. Python has also been ported to the Java and .NET virtual machines.

2.4 Pygtk

PyGTK provides a convenient wrapper for the GTK+ library for use in Python programs, taking care of many of the boring details such as managing memory and type casting. When combined with the PyORBit, gnome-python, gnome-python-desktop or gnome-python-extras modules, it can be used to write full featured Gnome applications.

GTK+ is a GUI toolkit for developing graphical applications that run on POSIX systems such as Linux, Windows and MacOS X (provided that an X server for MacOS X has been installed). It provides a comprehensive set of widgets, and supports Unicode and bidirectional text. It links into the Gnome Accessibility Framework through the ATK library.

2.5 Artificial Neural Network

An artificial neural network (ANN) or commonly just neural network (NN) is an interconnected group of artificial neurons that uses a mathematical model or computational model for information processing based on a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network.

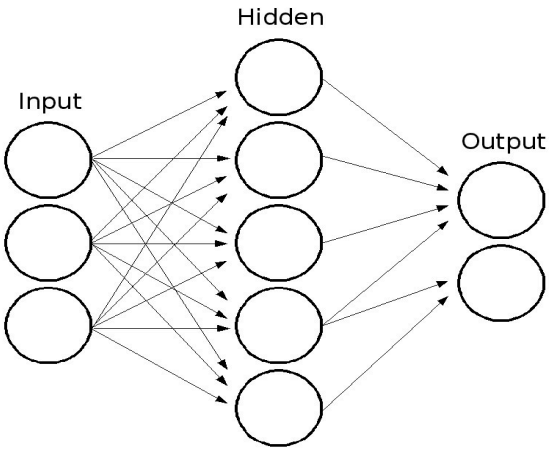


Fig 2.1 ANN model

The architecture of a Perceptron consists of a single input layer of many neurodes, and a single output layer of many neurodes. The simple "networks" illustrated at the beginning, to produce logical "AND" and "OR" operations have a Perceptron architecture. But to be called a Perceptron, the network must also implement the Perceptron learning rule for weight adjustment. This learning rule compares the actual network output to the desired network output to determine the new weights. For example, if the network illustrated gives a "0 1 0" output when "0 1 1" is the desired output for some input, all of the weights leading to the third neurode would be adjusted by some factor.

2.6 Unicode

Unicode is an industry standard designed to allow text and symbols from all of the writing systems of the world to be consistently represented and manipulated by computers. Developed in tandem with the Universal Character Set standard and published in book form as The Unicode Standard, Unicode consists of a character repertoire, an encoding methodology and set of standard character encodings, a set of code charts for visual reference, an enumeration of character properties such as upper and lower case, a set of reference data computer files, and rules for normalization, decomposition, collation and rendering.

2.7 Parsing

Parsing is the process of analyzing an input sequence (read from a file or a keyboard, for example) in order to determine its grammatical structure with respect to a given formal grammar. It is formally named syntax analysis. A parser is a computer program that carries out this task. The name is analogous with the usage in grammar and linguistics. The term parseable is generally applied to text or data which can be parsed.

Parsing transforms input text into a data structure, usually a tree, which is suitable for later processing and which captures the implied hierarchy of the input. Generally, parsers operate in two stages, first identifying the meaningful tokens in the input, and then building a parse tree from those tokens.

2.8 Free Software

As Richard Stallman says, Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission.

Chapter 3

Methodology

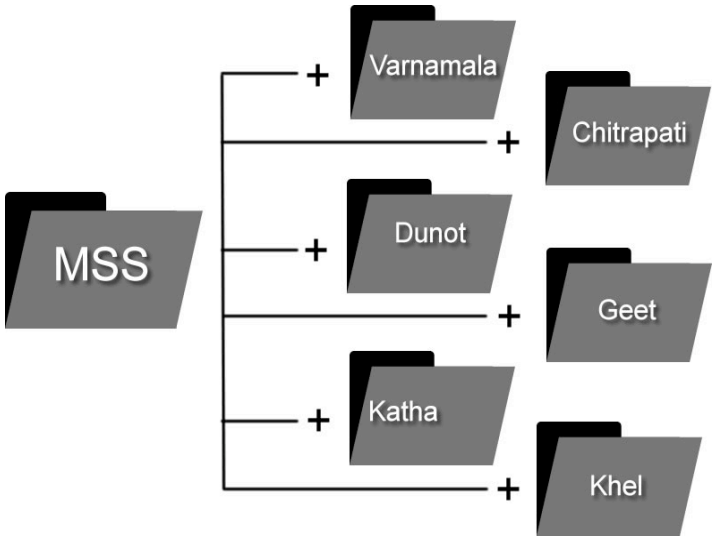
In this chapter we discuss about the user interface of MSS and put light on each modules which comprise MSS.

3.1 User Interface

As already stated MSS has six modules , or MSS could be seen as the combination of six different software components. These software component could be briefly explained as following:

- Varnamala – for recognizing alphabets and their order, read and write them
- Dunot – for knowing order of numbers and learning simple hand calculation
- Katha - for reading and writing story or, preparing smaller static presentation
- Chitrapati – for freehand drawing, learning basic geometry and alignments of shapes and coordinates
- Geet – for listening nursery rhymes
- Khel – for playing simpler educational games

Each module behaves differently, hence to engineer this we devised the MSS class on top of other classes and gave it a interface with link to all module. The class arrangement of MSS could be visualized on figure below.



The agile modeling has been followed to make the system acceptable to evolution. At each part, we started building component by studying the user requirements from secondary data collected, which included the Nepal Government curriculum of class 1 and constructionists tools implemented by IFCD. Then the mock ups were developed in sketch. After designing the activity diagram, the prototypes were built and coding was done at the end. After review of each component the whole process was again repeated by adjusting major or minor component, reconstituting new components and deprecating

some old components. The process model could be represented by evolutionary model also known a spiral model.

In this model the software is developed in a series of evolutionary releases. During early iterations, the release might be a paper model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

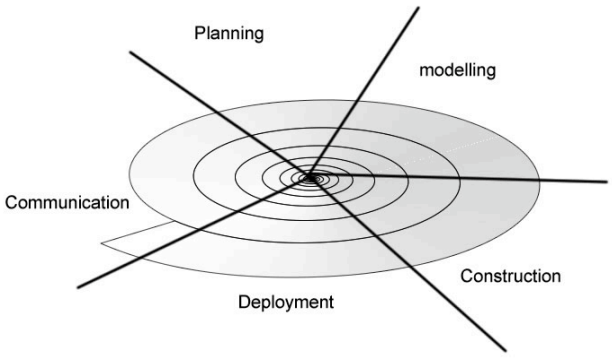


Fig 3.2 Evolutionary Spiral model

Since the system is targeted for children with less or no education, it must involve as simple interaction as possible. But at the same time MSS has multiple modules, which certainly makes the user interaction complex, vague and complicated, the normalization of user interaction could be done only by presenting the interaction and interface both in modular approach. The first main interaction provided for user can propagate three signal, one at the time when user chooses one module, next when user changes the module and another at the time user leaves the module. These factors lead too less to do with main interface interaction making it completely simple. Virtually user just chooses or exits the module at this level. So explanation of each module's user interaction could be provided separately, but representing them collaboratively.

At selection of Varnamala module, the user has three ways in which he can interact with the system. The user can select the component in one sub-module , reorder components in another module or draw shapes in next module, here too all sub-modules are placed separately for simplicity.

Similarly the Dunot module also works on the basis of three interaction with the user from its front end. Once initiated user could choose/view the problems, next signal occurs when user gives answer to the problem. The signal could be one of the type, either user selects the solution or write the solution.

In Katha module, user is presented with two interfaces, but cleverly only one at a time without much notice of user. In each interface the user interacts to system with the help of two signals. In story reading module the two signals occur when user chooses the story or while user navigates the story. In story writing module the two signals are when user

writes scripts an uploads for each slide or when user saves the story. These interfaces have been made as simpler as possible.

Chitrapati module also contains three simple ways for user to interact with the system. The first signal occurs when user writes/executes commands or controls. Another signal occurs when user opens shapes and the third signal occurs when user freehand draws the canvas. All this three signal have direct effect on canvas and they can occur at any random sequence until the module is not left or change.

Geet and Khel module presents somewhat similar interaction interfaces to the user, in both module user is offered only two signal to interact. The first signal occurs when user selects one of the song/game and another occurs when user exits the game.

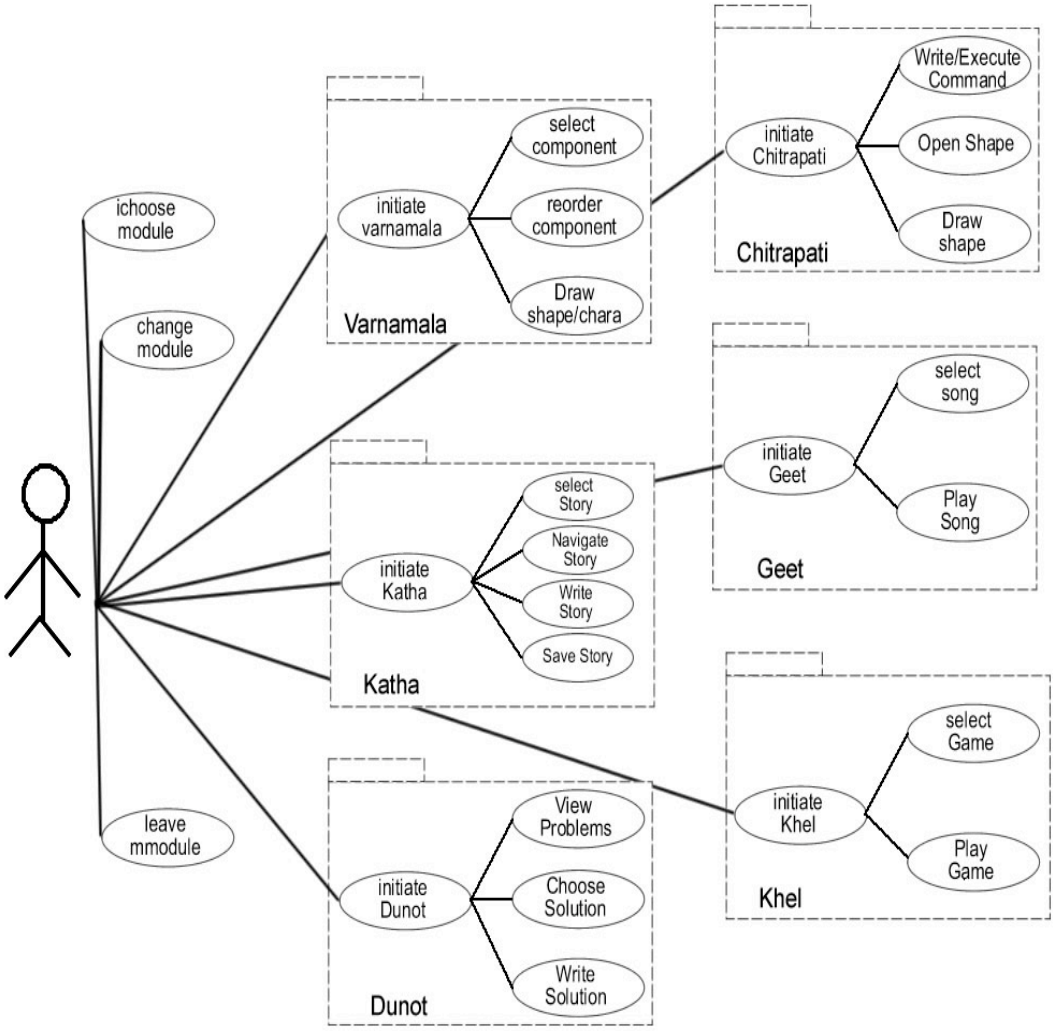


Fig 3.3 Use case diagram of MSS

3.2 Varnamala

Reading and writing are the foundations for learning processes. And alphabets are the foundations for reading and writing. Even if learning alphabets is not the beginning of learning process it does play the role of a milestone in learning path. It is thought that learning alphabets marks the beginning of formal learning process. “Varnamala” means the collection of alphabets in Nepali. The main motive of this module is intended to help children to learn recognizing and writing alphabets. It is divided into three activities:

3.2.1 Identification: This activity helps children learn and identify the alphabets. At first they are made familiar with different shapes, which form the basics of alphabets. It is done like a game where they need to select similar or dissimilar shapes from a group of shapes. Later the same game is played with alphabets in place of shapes so that they get familiarized with alphabet structures at least. It needs only clicking skill.

3.2.2 Arrangement: This activity helps children to learn ordering and arrangements after they get familiarized with the alphabets. Here they are provided with random pattern from which they need to rearrange the shapes by drag and drop. The game of arrangement is played with shapes and later the shapes are replaced with the alphabets. At the end of this activity children will be familiar with alphabets and their order.

3.2.3 Writing: Here, children learn to write the alphabets. At first they are provided with different shapes to draw. And their progress is tracked with the use of Artificial Neural Network. When they start drawing the basic shapes well, they are provided with the interface to draw alphabets. At the end of this part of Varnamala, children will be able to write the alphabets and draw different shapes.

The methodology used to achieve the Varnamala can be well explained with the class diagram below:

Varnamala	
Attributes:	
1. Shapes	1. Stores images of different basic shapes
2. Alphabets	2. Stores standard alphabet images
3. Canvas	3. Stores the drawings drawn on the active screen
4. Pixbuff	4. Stores the information of the current canvas as an image
5. Weight	5. Stores weight of the ANN training set
6. Pixmap	6. Stores the information of the currently dragged image
Operations:	
1. draw_random()	1. Draws the images in random pattern in screen
2. get_input()	2. Gets the current user input and checks it
3. drag_drop()	3. Handles the drag and drop mechanism
4. get_status()	4. Gets the current status of the screen for result comparing
5. save_pixmap()	5. Saves the Pixbuf value as image variable
6. filter_drawing()	6. Processes the drawing to be passed to ANN
7. compare_results()	7. Compare the drawing with standard pattern
8. show_results()	8. Displays the results/ progress.

The activity diagram of the Varnamala is as follows:

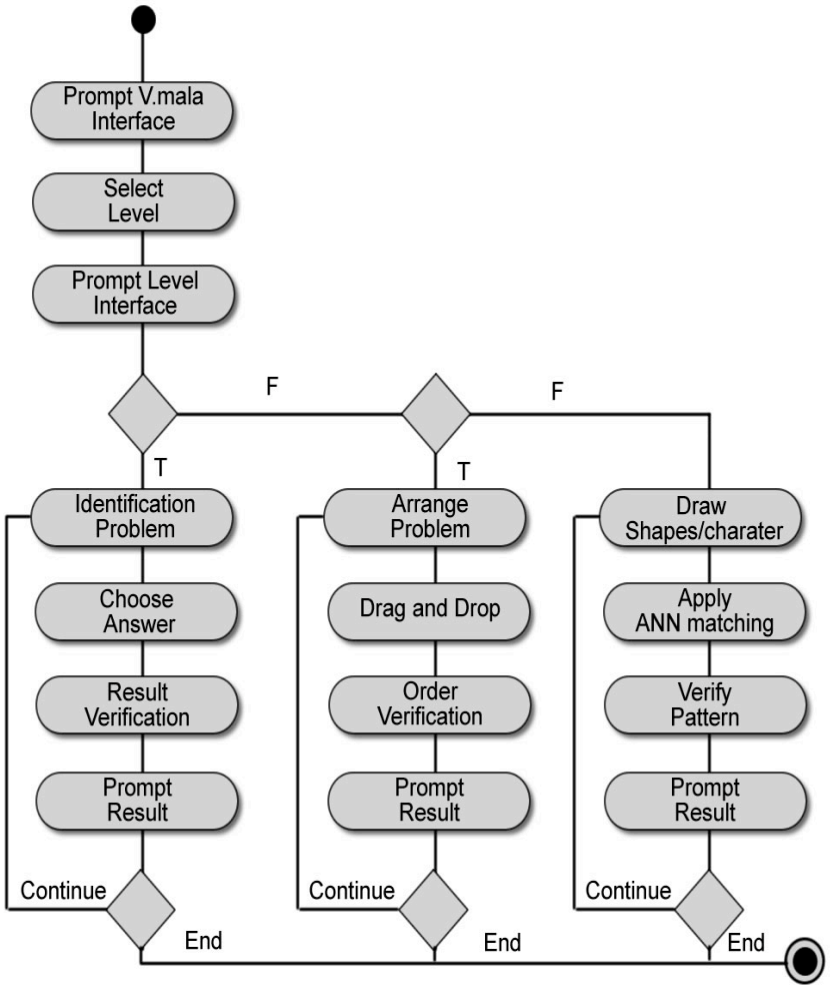


Fig 3.4 Activity Diagram of Varnamala

3.3 Dunot

"Dunot" is a common word related to basic mathematics in Nepali, meaning arranged table of numbers as in multiplication table. Dunot module has been designed to assist children in learning basic mathematical skills. Here they learn basic addition, subtraction and ordering of numbers. Now it has two activities:

3.3.1 Arrangement: In this activity children learn about different kind of number arrangements. Currently we have included greater than and less than ordering in first stage and then ascending and descending ordering.

3.3.2 Calculation: Here children learn basic addition and subtraction skills. At first they learn single digit addition and subtraction, then carry-over additions and subtractions.

The methodology used to achieve the Dunot can be well explained with the class diagram below:

Dunot	
Attributes: 1. Level 2. Result 3. User_input	 1. Stores current level of the children 2. Stores the result of the current problem 3. Stores user input to be checked with result
Operations: 1. show_question() 2. check_input() 3. change_level()	 1. Displays question of current level to the user 2. Checks the user input with the answer 3. Changes the level when current level is passed

The activity diagram of the Varnamala is as follows:

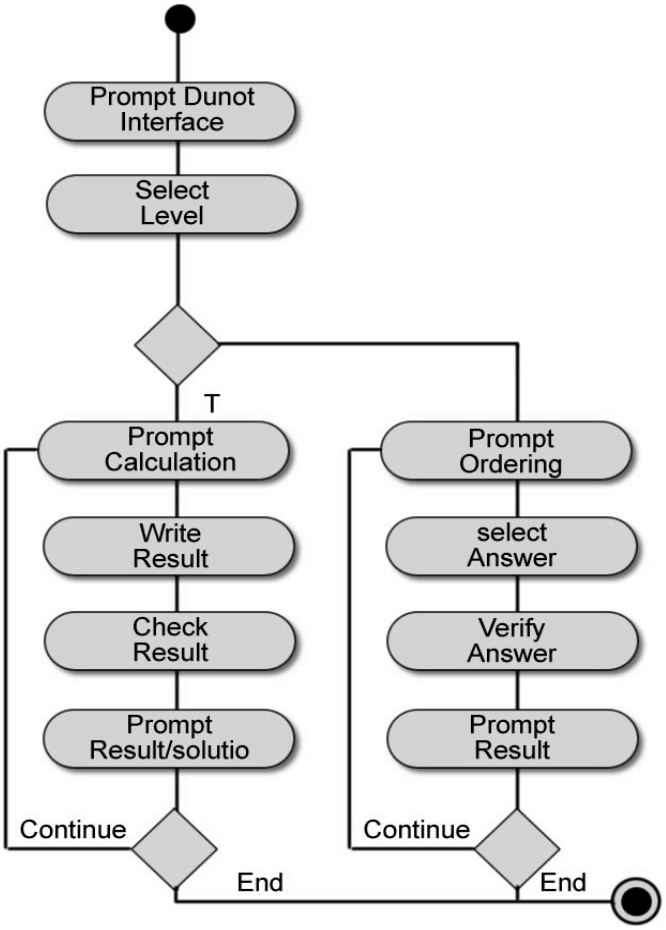


Fig 3.5 Activity Diagram of Dunot

3.4 Katha

Katha a storyteller interface is composed of two distinct components to serve two purposes. The children can learn standard stories and they could also prepare story of their own. The story here is a collection of image files with a standard file having information at about script, scene, and writer's information. These files are presented in a neat interface with easily navigable controls. The two components are Read story and Write story.

3.4.1 Read Story: This interface is starting interface for katha module, User can either open a story, pause katha module by minimizing, write new story or exit the module. Once a story is opened the information about story like title, writer's name, email etc. are Displayed on the interface. The user can navigate the story by using controls for left and right navigator. User can restart same story by reloading at any time while reading story.

3.4.2 Write Story: When user enters this module from read story, readstory is temporarily paused, then User is presented with interface to write the Name of story and Writer's information. After that the user saves story , this creates file with story information and folder to store images needed in stories. Then the interface for writing story is presented which contains placeholder for image and textbox for writing story script under that image. When user finishes writing up the story by updating multiple files and exits story other necessary information are written as file footer.

The katha class is as follows:

Katha	
Attributes:	
1. Kathafile 2. Imagename 3. Slidecounter 4. Kathainfo 5. Script 6. Level	1. file containing the Story 2. image file for particular slide 3. counter to specify current slide number 4. information of the Story 5. Script for the specific slide 6. Story Level
Operations:	
1. Read_story() 2. Write_story() 3. Open_story()	1. display interface for reading Story 2. displays interface for writing story 3. displays available stories to choose

The Activity Diagram for katha module is given in next page:

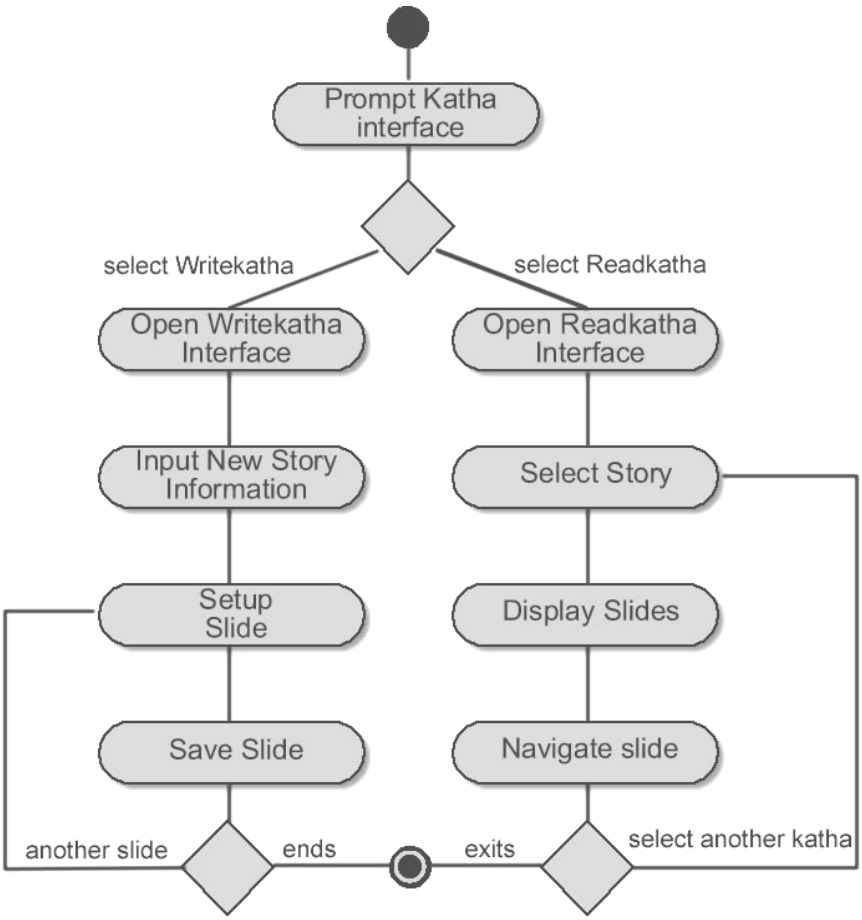


Fig 3.6 Activity Diagram of Katha

This katha interface could also be easily used for giving children information on different topics of general knowledge.

3.5 Chitrapati

Literally Chitrapati is Canvas, but being digital canvas it offers much more than physical canvas. This fact eplains that the scope of chitrapati is much wide, we said that chitrapati could be used to learn geometric shape, object alignment as well as creative freehand drawing. These three explanations fit in three component of Chitrapati. Yet all three are included into same interface of canvas. This is pretty pleasant experience to learn all things in a single environment. The three components are Freehand Drawing, Shape objects and command execution. Main interface contains link to this three components as drawing tools with a drawing canvas.

3.5.1 Freehand Drawing: In this method mouse or write-pad could be used as input device, once selecting this mode until you change to other modes user can

draw by clicking muse left button or dragging stylus over the pad. The object is drawn under current coordinate of mouse ar stylus in the canvas.

3.5.2 Shape Object: The custom shape could be stored in a file either as pixmaps sequence or the sequential script. This files are retrival from any point at the canvas. If user opens this shape files then starting from current coordinate point the object are drawn on the canvas, this method helps to align shapes at different place and draw complex shape by including patterns of shapes drawn earlier.

3.5.3 Command Execution: This model is somewhat inspired by children programming language logo. In this model the current cursor point is visible and set of commands are available which helps to draw the different shapes in canvas by moving the cursor which leaves its trail while moving. Few commands are:

- Left <value> - rotate left by <vlue> degree
- Right <value> - rotate right by <value> degree
- Forward <value> - moves forward by <value> in current direction
- Penup - doesn't draw until next pendown command
- Pendown – ends last penup command
- Loop <value> [command1,command2] – loops commands inside bracket by value times

The Chitrapati class is as follow:

Chitrapati	
Attributes: 1. Current method 2. Canvas info 3. Cursor pos 4. Current angle 5. Value 6. Command	 1. stores current drawing method 2. holds canvas info for saving 3. records cursor position 4. stores current angle 5. stores value from command 6. stores command
Operations: 1. Display_chitra() 2. Select_method() 3. Draw_freehand() 4. Draw_shape() 5. Run_script() 6. Update_canvas() 7. Save_chitra() 8. Exit_chitra()	 1. brings up the interface 2. selects drawing method 3. draws freehand on canvas 4. provides interface for shape opening 5. prompts command to draw 6. records canvas information after each steps 7. saves canvas contains to file 8. quits chitrapti module

The activity diagram for chitrapati is as follow:

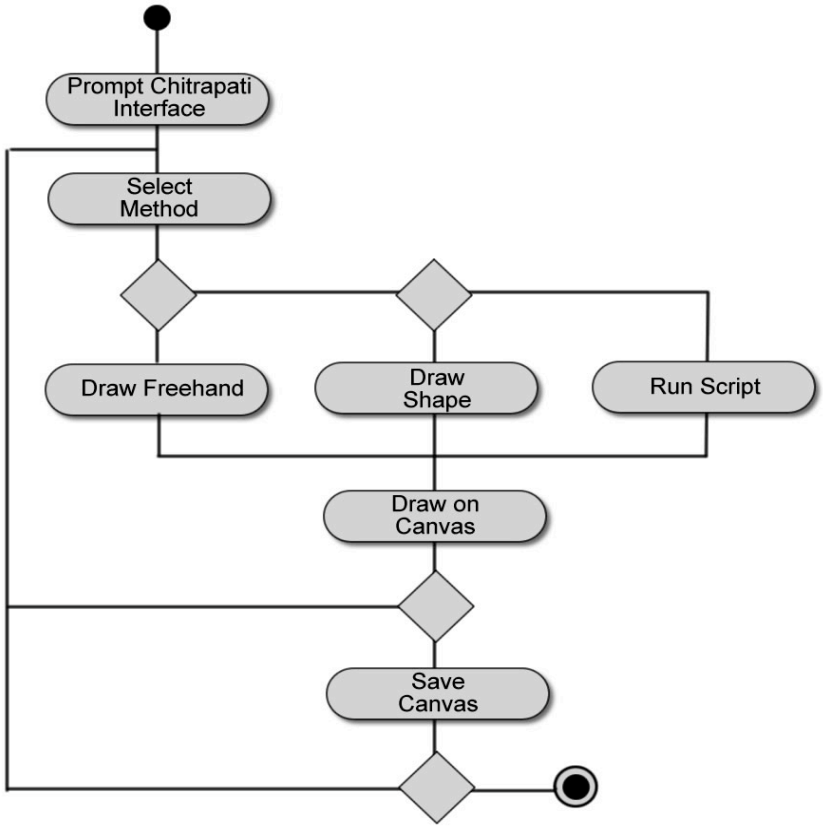


Fig 3.7 Activity Diagram of Katha

3.6 Geet

Nursery Rhymes have great effect in learning. The rhythm and words in rhymes are moral guide for children. MSS also includes geet module which is rather simpler than other modules but with greater effectiveness. MSS geet module doesn't have sub modules, Rather it contains simple engine for playback purpose only and provide user with very simple interface to choose and play songs.

When the interface is opened user is presented with the icons of song to select from , after that the is played and simple control like pause and stop are shown. The user can recursively play the songs or leave the module. The geet module could also be regarded as one of the important entertainment module.

The Class Diagram for Geet module is

Geet	
Attributes: 1. Songname 2. songmode 3. listsong	1. name of the song to play 2. current mode of song 3. list of available song
Operations: 1. display_geet()	1. display Geet interface

Geet	
2. select_geet() 3. play_geet() 4. pause_geet() 5. stop_geet() 6. exit_geet()	2. select chosen song for playing 3. Engine to play song 4. pause current playing song 5. stop playing song 6. exits geet module

The activity Diagram for Geet module is:

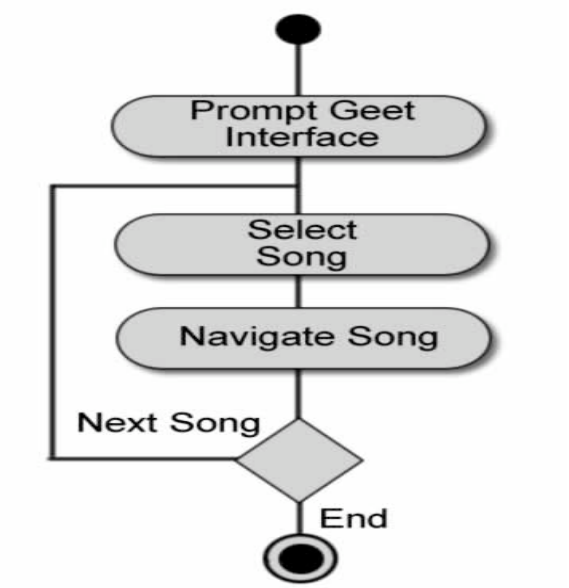


Fig 3.8 Activity Diagram of Geet

3.7 Khel

Khel is another entertaining cum educational module, this module also somewhat functions like song module. Instead of being a game by itself khel is just a frame to support collection of individual python based games. Khel maintains a log file with information of all available games within the module. similar to geet it present user with all available games as a selection icons. when user chose the game the control is transferred to the python file describing that game. when user exits or quit game then user is again presented with main interface of game.

Khel	
Attributes: 1. gamename 2. gamescript 3. gamelist 4. mode	1. name of the game to play 2. list with script of game 3. list of available game 4. current mode
Operations: 1. display_game() 2. select_game() 3. play_game() 4. exit_game()	1. display Game interface 2. select chosen game for playing 3. transfer controls to play game 4. exits game module

The activity diagram for game module is

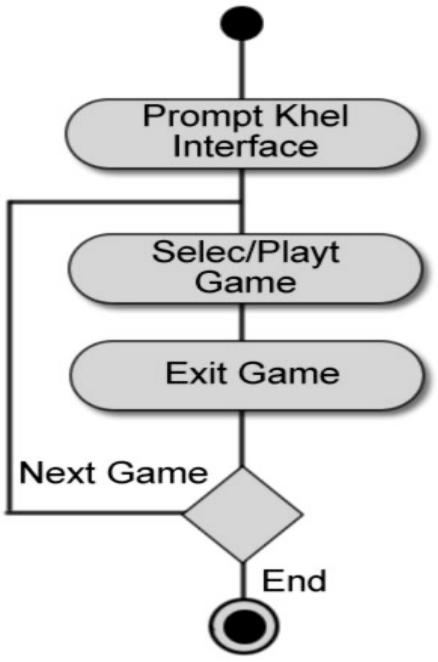


Fig 3.9 Activity Diagram of Khel

Chapter 4

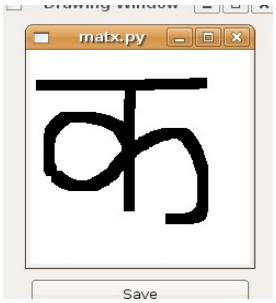
Analysis

In this chapter we have put the analysis part of some components that have been used in the project.

4.1 character recognizer ANN

The character recognizer was featured on Varnamala module of the MSS. The working establishment and performance measure are analyzed in following section. The Artificial Neural Network with 81 input nodes and single hidden layer with 165 nodes was used. for brevity of calculation and recognition the training input was used to assign the weight and save the weight in file in seperate program at the time of recognition the file with weight was opened to initialize the weight to nodes and calculation was done. The whole schematic is explained below.

4.1.1 Training data preparation: Training data were prepared by converting the standar draw area drawn part in 81 small components and counting the frequency of drawn pixel in each components and converting it into list.



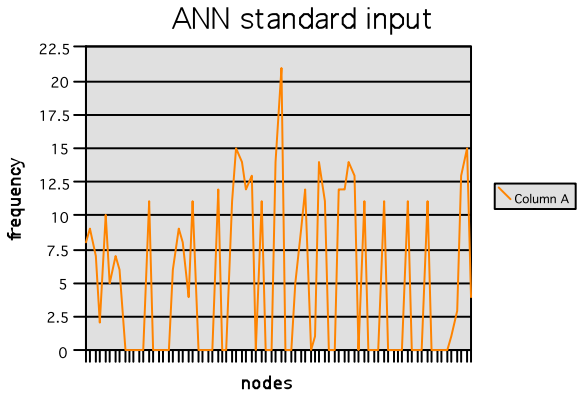


Fig 4.1 Input character preparation

4.1.2 Neural Network: we already know that the neural network contains 81 input nodes and 165 single hidden layer nodes, we have used only 5 output node for brevity of calculation as saving weight to file allows flexibility in recognizing more than 5 output using multiple weight files. The calculation formulas are as follows.

a. Calculating weight to hidden node

```
sum=sum+wkj[k][j]*charac[c][k]
a[j]=1.0/(1.0+math.exp(-sum))
```

b. Calculating output nodes weight

```
sum1=sum1+wji[j][m]*a[j]
O[c][m]=1.0/(1.0+math.exp(-sum1))
```

c. Backpropagation of weight

```
deltawji=alpha*a[j]*(T[c][i]-O[c][i])*O[c][i]*(1.0-O[c][i])
wji[j][i]=wji[j][i]+deltawji

sum = sum +(a[i]*(1.0-a[i]) *(T[c][i]-a[i]) * wji[j][i])
deltawkj=alpha * charac[c][k] * a[j] * (1.0-a[j]) * sum
wkj[k][j]=wkj[k][j]+deltawkj
```

d. Recognition

```
for i in range(5):
      if abs(On[i]) > abs(sum1):
            sum1=On[i]
            mm=i
character=mm
accuracy=On[mm]*100
```

4.1.3 Result: The two sample was taken to test the result, and the samples and their frequency were as follow

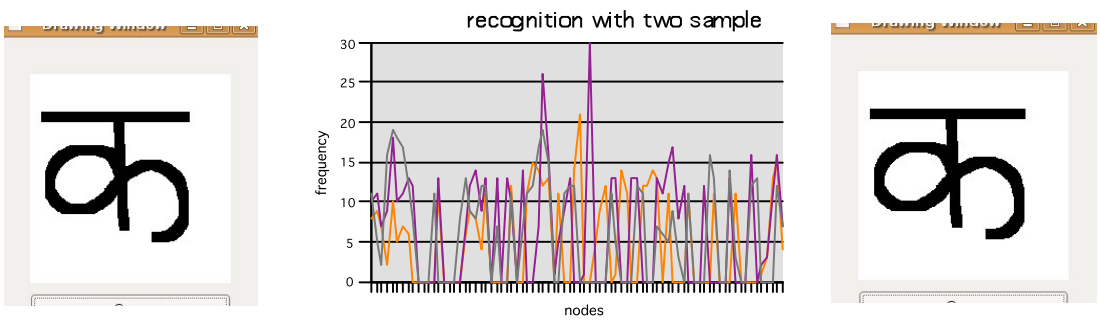


Fig 4.2 Input character recognition

The result was that first sample was recognized with 96.497% of accuracy and second sample was recognized with 86.225% of accuracy.

Chapter 5

Conclusion

We started the project with the objective of transformation of society by transforming the education with the use of ICT. By this point we have achieved so much of success in our objective. We have teamed up with many organizations, individuals and schools towards the use of ICT and constructivism in education for the transformation of education. Even if our technical works can be taken as negligible, it is our pleasure to say that our works will be creating an extra domain in engineering and ICT studies and implementation.

Appendix

A. Source Code Snippets

A.1 NN Engine

```
""" Weight initialization start"""
for cnt in xrange(165):
      a.append(0.0)

for k in xrange(5):
      for j in xrange(5):
            wj.append(0.0)
      T.append(wj)
      wj=[]

for k in xrange(5):
      for j in xrange(5):
            wj.append(0.0)
      O.append(wj)
      wj=[]

for k in xrange(5):
      for j in xrange(5):
            wj.append(0.0)
      error.append(wj)
      wj=[]

for k in xrange(81):
      for j in xrange(165):
            rnd=r.rand(100)
            wj.append((50.0-rnd)/100.0)
      wkj.append(wj)
      wj=[]
l=len(wkj)

for j in xrange(165):
      for i in xrange(5):
            rnd=r.rand(100)
            wj.append((50.0-rnd)/100.0)
      wji.append(wj)
      wj=[]
l=len(wji)

""" weight initialization complete"""

""" Training Network by giving supervised data"""
n=1000
while n>0 and abs(maxerror)>0.05:
      print "at step"+str(1000-n)+"  "+str(maxerror)

      n=n-1
      maxerror=0.0
```

```
      for c in xrange(charlen):
            for j in xrange(165):
                  sum=0.0
                  for k in xrange(81):
                        sum=sum+wkj[k][j]*charac[c][k]
                        a[j]=1.0/(1.0+math.exp(-sum))

            for m in xrange(charlen):
                  sum1=0.0
                  for j in xrange(165):
                        sum1=sum1+wji[j][m]*a[j]

                  O[c][m]=1.0/(1.0+math.exp(-sum1))
                  if m==c:
                        T[c][m]=1.0
                  else:
                        T[c][m]=0.0
                  error[c][m]=T[c][m]-O[c][m]
                  if abs(error[c][m]) > abs(maxerror):
                        maxerror=error[c][m]


            """ Backpropagating errors """

            for i in xrange(charlen):
                  for j in xrange(165):
                        deltawji=alpha*a[j]*(T[c][i]-
                              O[c][i])*O[c][i]*(1.0-O[c][i])
                        wji[j][i]=wji[j][i]+deltawji

            for k in xrange(81):
                  for j in xrange(165):
                        sum=0.0
                        for i in xrange(charlen):
                              sum = sum +(a[i]*(1.0-a[i]) *(T[c][i]-
                              a[i]) * wji[j][i])
                              deltawkj=alpha * charac[c][k] * a[j] *
                              (1.0-a[j]) * sum
                              wkj[k][j]=wkj[k][j]+deltawkj

""" Training Network ends """#
```

A.2 NN Recognizer

```
for cnt in range(5):
      On.append(0.0)
print On
charlen=5
for j in xrange(165):
      sum=0.0
      for k in xrange(81):
```

```
            sum=sum+wkj[k][j] * chara[k]
            a[j]=1.0/(1.0+math.exp(-sum))

for mm in range(5):
      sum1=0
      for j in xrange(165):
            sum1=sum1+wji[j][mm]*a[j]
      On[mm]=1.0/(1.0+math.exp(-sum1))

sum1=On[0]
mm=0

for i in range(charlen):
      if abs(On[i]) > abs(sum1):
            sum1=On[i]
            mm=i
character=mm
accuracy=On[mm]*100#
```

A.3 Drawing

```
def configure_event(widget, event):
      global pixmap
      x, y, width, height = widget.get_allocation()
      pixmap = gtk.gdk.Pixmap(widget.window, width, height)
      pixmap.draw_rectangle(widget.get_style().white_gc,
                            True, 0, 0, width, height)
      return True

# Redraw the screen from the backing pixmap
def expose_event(widget, event):
      x , y, width, height = event.area
      widget.window.draw_drawable(widget.get_style().fg_gc[gtk.STATE_NOR
MAL],
                                  pixmap, x, y, x, y, width, height)
      return False

# Draw a rectangle on the screen
def draw_brush(widget, x, y):
      rect = (x - 5, y - 5, 10, 10)
      pixmap.draw_rectangle(widget.get_style().black_gc, True,
                            rect[0], rect[1], rect[2], rect[3])
      xyd=[x,y]
      global picbuf
      picbuf.append(xyd)
      widget.queue_draw_area(rect[0], rect[1], rect[2], rect[3])

def button_press_event(widget, event):
      if event.button == 1 and pixmap != None:
            draw_brush(widget, event.x, event.y)
      return True

def motion_notify_event(widget, event):
      if event.is_hint:
            x, y, state = event.window.get_pointer()
      else:
```

```
            x = event.x
            y = event.y
            state = event.state
      if state & gtk.gdk.BUTTON1_MASK and pixmap != None:
            draw_brush(widget, x, y)
      return True#
```

A.4 Drag and Drop

```
def __init__(self):
      self.btnid = ["ka", "kha", "ga"]
      self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
      self.window.connect("delete_event", self.delete_event)
      self.window.set_border_width(20)

      table = gtk.Table(6, 6, True)
      self.window.add(table)

      btn1 = gtk.Button(self.btnid[0])
      btn1.connect("drag_data_get", self.sendCallback)
      btn1.drag_source_set(gtk.gdk.BUTTON1_MASK,
      self.fromButton,gtk.gdk.ACTION_COPY)


      table.attach(btn1,  0, 1, 0, 1)

      btn2 = gtk.Button(self.btnid[1])
      btn2.drag_dest_set(gtk.DEST_DEFAULT_MOTION |
      gtk.DEST_DEFAULT_HIGHLIGHT | gtk.DEST_DEFAULT_DROP,
      self.toButton, gtk.gdk.ACTION_COPY)
      btn2.connect("drag_data_received", self.receiveCallback)

      table.attach(btn2,  5, 6, 5, 6)

      self.window.show_all()

      def sendCallback(self, widget, context, selection, targetType,
      eventTime):
            selection.set(selection.target, 8, widget.get_label())

      def receiveCallback(self,  widget, context,  x,  y, selection,
      targetType, time):
            widget.set_label(selection.data)
            print selection.data + str(x) + " " + str(y) + " copied"
```

A.5 Chitrapati snippet

```
def on_button1_clicked(self, widget):
            self.inn=self.wTree.get_widget("entry1")
            self.draw=self.wTree.get_widget("drawingarea1")
            self.draw.set_size_request(400,400)
            drawable=self.draw.window
            color = gtk.gdk.Color(red=65551, green=255, blue=6998,
            pixel=5)
            gc = drawable.new_gc(foreground=color, background=None)
            #drawable.draw_line(gc,10,10,40,40)
```

```
            #drawable.draw_rectangle(gc,1,10,10,40,40)
            txt=self.inn.get_text()
            cmd=string.split(txt)
            global mx
            global my
            global degree
            global theta
            value=int(cmd[1])
            pi=3.141592652654
            print cmd[0]
            print value
            theta=0
            if cmd[0]=="rotate":
                  degree=degree+value

            if cmd[0]=="fwd":
                  theta=(pi/180.0)*degree
                  si=math.sin(theta)
                  co=math.cos(theta)
                  x=int(value * si)
                  y=int(value * co)
                  drawable.draw_line(gc,mx,my,mx+x,my+y)
                  mx=mx+x
                  my=my+y

      def   on_button2_clicked(self, widget):
            self.draw=self.wTree.get_widget("drawingarea1")
            self.draw.set_size_request(400,400)
            drawable=self.draw.window
            color = gtk.gdk.Color(red=0, green=0, blue=0, pixel=5)
            gc   =   drawable.new_gc(foreground=color,   background=color)
      chooser=gtk.FileChooserDialog(title=None,action=gtk.FILE_CHOOSER_A
CTION_OPEN,
buttons=(gtk.STOCK_CANCEL,gtk.RESPONSE_CANCEL,gtk.STOCK_OPEN,gtk.RESPONS
E_OK))
            chooser.set_default_response(gtk.RESPONSE_OK)
            response = chooser.run()
            fname=chooser.get_filename()
            chooser.destroy()
            f=file(fname,"r")
            global mx
            global my
            d=f.readline()
            if d=="start":
                  drawable.draw_line(gc,mx,my,mx,my)
            c=1
            while c!=0:
                  txt=f.readline()
                  cmd=string.split(txt)
                  value=int(cmd[1])
                  print cmd[0]
                  print value
                  if cmd[0]=="start":
                        drawable.draw_line(gc,mx,my,mx,my)
                  if cmd[0]=="right":
                        drawable.draw_line(gc,mx,my,mx+value,my)
                        mx=mx+value
```

```
                if cmd[0]=="left":
                      drawable.draw_line(gc,mx,my,mx-value,my)
                      mx=mx-value
                if cmd[0]=="down":
drawable.draw_line(gc,mx,my,mx,my+value)
                      my=my+value
                if cmd[0]=="up":
                      drawable.draw_line(gc,mx,my,mx,my-value)
                      my=my-value
                if cmd[0]=="end":
                      c=0#
```

B. Olpc Hardware

Physical dimensions:

Dimensions: 193mm × 229mm × 64mm (as of 3/27/06—subject to change)
Weight: Less than 1.5 KG (target only—subject to change)
Configuration: Convertible laptop with pivoting, reversible display; dirt- and moisture-resistant system enclosure

Core electronics:

CPU: AMD Geode GX-500@1.0W(datasheet)
CPU clock speed: 366 Mhz
Compatibility: X86/X87-compatible
Chipset: AMD CS5536 South Bridge (datasheet)
Graphics controller: Integrated with Geode CPU; unified memory architecture
Embedded controller (for production), ENE KB3700: Image:KB3700-ds-01.pdf
DRAM memory: 128MB dynamic RAM
Data rate: Dual – DDR266 – 133 Mhz
BIOS: 1024KB SPI-interface flash ROM; LinuxBIOS open-source BIOS; Open Firmware bootloader
Mass storage: 512MB SLC NAND flash, high speed flash controller
Drives: No rotating media

Display:

Liquid-crystal display: 7.5” Dual-mode TFT display
Viewing area: 152.4 mm × 114.3 mm
Resolution: 1200 (H) × 900 (V) resolution (200 dpi)
Mono display: High-resolution, reflective monochrome mode
Color display: Standard-resolution, quincunx-sampled, transmissive color mode

Integrated peripherals:

Keyboard: 70+ keys, 1.2mm stroke; sealed rubber-membrane key-switch assembly
Keyboard Layouts
Layout pictures - US International, Thai, Arabic, Spanish, Portuguese, Nigeria
Cursor-control keys: five-key cursor-control pad; four directional keys plus Enter
Touchpad: Dual capacitance/resistive touchpad; supports written-input mode
Audio: Analog Devices AD1888, AC97-compatible audio codec; stereo, with dual internal speakers; monophonic, with internal microphone and using the Analog Devices SSM2211 for audio amplification
Wireless: Marvell 88W8388, 802.11b/g compatible; dual adjustable, rotating coaxial antennas; supports diversity reception
Status indicators: Power, battery, WiFi; visible lid open or closed
Video camera: 640x480 resolution, 30FPS

External connectors:

Power: 2-pin DC-input, 10 to 25 V, -23 to -10 V
Line output: Standard 3.5mm 3-pin switched stereo audio jack
Microphone: Standard 3.5mm 2-pin switched mono microphone jack; selectable sensor-input mode
Expansion: 3 Type-A USB-2.0 connectors; SD Card slot
Maximum power: 500 mA (total)

Battery:

Pack type: 5 Cells, 6V series configuration
Fully-enclosed “hard” case; user removable
Capacity: 22.8 Watt-hours
Cell type: NiMH
Pack protection: Integrated pack-type identification
Integrated thermal sensor
Integrated polyfuse current limiter
Cycle life: Minimum 2,000 charge/discharge cycles (to 50% capacity of new, IIRC).
Power Management will be critical

BIOS/loader:

LinuxBIOS is our BIOS for production units; Open Firmware is used as the bootloader.
Environmental specifications:

Temperature: somewhere in between typical laptop requirements and Mil spec; exact values have not been settled
Humidity: Similar attitude to temperature. When closed, the unit should seal well enough that children walking to and from school need not fear rainstorms or dust.
Maximum altitude: -15m to 3048m (14.7 to 10.1 psia) (operating), -15m to 12192m (14.7 to 4.4 psia) (non-operating
Shock 125g, 2ms, half-sine (operating) 200g, 2ms, half-sine (non-operating)
Random vibration: 0.75g zero-to-peak, 10Hz to 500Hz, 0.25 oct/min sweep rate (operating); 1.5g zero-to-peak, 10Hz to 500Hz, 0.5 oct/min sweep rate (nonoperating)
2mm plastic walls (1.3mm is typical for most systems).

C. Olpc Software

Linux Kernel

For the main kernel, we are using the Fedora Rawhide version of the Linux kernel, which means that we are tracking the main kernel fairly closely. The OLPC specific bits of the Rawhide kernel are pulled from the olpc-2.6 GIT tree on dev.laptop.org:http://dev.laptop.org/git.do?p=olpc-2.6;a=summary

Programming environments

Python version 2.4 or, more likely, version 2.5
Javascript
CSound, our sound and music environment
Squeak / Etoys, a media-rich authoring environment (Please see Sugar_EToys for a detailed description of the Sugar implementation.)
Libraries and Plugins
Tinymail (Possible, still looking for design ideas.)
Mozilla Gecko/Xul
GUI toolkit (GTK+) (Gnome)
Sugar (UI) RedHat/OLPC/Pentagram
Pango text layout
Gnome Accessibility toolkit (ATK)
Python GTK+ bindings version 2.10
Cairo 2D-graphics support
X Window System X.org Foundation
Fedora Linux
Font rendering (Freetype)
Avahi local service discovery
Multimedia framework: gstreamer and RealNetworks

D. Sugar

Here is a short tutorial about writing a sugar activity.

Write the build system
Create the autogen.sh script
You should only need to change the module name and activity file from this example.

```
#!/bin/sh
# Run this to generate all the initial makefiles, etc.

srcdir=`dirname $0`
test -z "$srcdir" && srcdir=.

PKG_NAME="sugar-drawing"

(test -f $srcdir/drawing.activity) || {
    echo -n "**Error**: Directory "\`$srcdir\'" does not look like the"
```

```
    echo " top-level $PKG_NAME directory"
    exit 1
}

which gnome-autogen.sh || {
    echo "You need to install gnome-common from the GNOME CVS"
    exit 1
}

REQUIRED_AUTOMAKE_VERSION=1.9 USE_GNOME2_MACROS=1 . gnome-autogen.sh
```

Create a configure.ac file

The first line of configure.ac specifies the name of the application, the version number, a bug reporting address, and the name of the module.

The configure.ac file contains a list of all the Makefiles in your project. You will need one Makefile for every directory in your project.

```
AC_INIT([Sugar Drawing],[0.1],[],[sugar-drawing])
AM_INIT_AUTOMAKE

AM_PATH_PYTHON

AC_OUTPUT([
Makefile
other_dir/Makefile
])
```

Other required files

Automake checks for the existence of a few required files, you have to create them even if empty:

```
$ touch NEWS README AUTHORS ChangeLog
```

Makefile.am

This file tells automake how to install your program.

in the Makefile.am, SUBDIRS is a space-separated list of subdirectories of the current folder, an activitydir that describes where files in the current folder will be installed, and activity_PYTHON which describes what files to install.

The backslash at the end of the activity_PYTHON lines indicates that more data exists on the following lines. (Note that the last line does not have a backslash).

```
SUBDIRS = foo bar
activitydir = $(datadir)/sugar/activities/drawing
activity_PYTHON =               \
        __init__.py             \
        DrawingActivity.py

EXTRA_DIST = drawing.activity
```

install-data-local:

```
    sugar-setup-activity $(srcdir)/drawing.activity
```

Write the activity code

Write a subclass of sugar.activity.Activity. It's a GtkWindow so you can use the "add" method to insert your own widgets. The source of DrawingActivity.py demonstrates it:

```
import gtk

from sugar.activity.Activity import Activity

class DrawingActivity(Activity):
        def __init__(self):
                Activity.__init__(self)
                button = gtk.Button('Drawing')
                self.add(button)
                button.show()
```

Or if you are adapting a full application, your activity could look something like this: (I found I needed to delete the app object explicitly)

```
import gtk

from sugar.activity.Activity import Activity
from my_module.my_app import my_app

class DrawingActivity(Activity):
        def __init__(self):
                Activity.__init__(self)

                app = my_app.my_app()
                self.add(app.some_widget)
                app.do_show()
                self.connect('destroy',self.do_quit, app)

        def do_quit(self, event, app):
                app.do_quit()
                del app
```

create __init__.py

This is the package initialization file. It can be blank.

```
$ touch __init__.py
```

drawing.activity

Include the necessary information about the activity.

```
[Activity]
name = Drawing
id = org.laptop.sugar.Drawing
python_module = drawing.DrawingActivity.DrawingActivity
default_type = _drawing_olpc._udp
show_launcher = yes
```

Build and install

Initialize the build system. The value of prefix depends on the path of sugar-jhbuild:

```
$ ./autogen.sh –prefix=[SUGAR-JHBUILD]/build
```

Build and install:

```
$ [SUGAR-JHBUILD]/sugar-jhbuild shell
$ make
$ make install
```

7. Photos

References:

Books:

[1] *Python in a Nutshell*
by Alex Martelli , 2003 US O'Reilly

[2] *Python 2.1 Bible*
by Dave Brueck, Stephen Tanner - 2001 New York, NY : Hungry Minds, Inc.

[3] *Neural Networks for Pattern Recognition*
by Nigrin, Albert Nigrin - 1993 , The MIT Press

[4] *Digital Image Processing*
by Rafal C. Gonzalez, Richard E, Woods - 2002, Prentice-Hall of India

[5] *Text Processing in Python*
by David Mertz - 2003 Addison-Wesley Professional

[6] *Software Engineering: A Practitioner's Approach*
by Roger S Pressman – 2005 , Mc Graw Hill

[7] *Uml Distilled: A Brief Guide to the Standard Object Modeling Language*
by Martin Fowler - 2004 , Addison-Wesley Professional

[8] *Being digital*
by Nicholas Negroponte -1995 , New York : Knopf

[9] *The Construction of Reality in the Child*
by Jean Piaget -1955, web edition @ marxist.org

[10] *A guide for Thesis Writing*
by Laxmi Narayan Choudary, 2004, Nepal Engineering College

[11] *Learning Theories*
by wikibooks contributors, 2006, wikibooks.org

[12] *Pygtk 2.0 Tutorial*
by John Finlay, 2005, pygtk.org

Web:

[1] One Laptop Per Child - http://laptop.org
[2] Free Software Foundation - http://fsf.org

[3] constructivism, jean piaget, semouyr papert, nicholas Negroponte, logo and most of the basics - http://wikipedia.org
[4] Squeak e-toys - http://squeak.org
[5] Ubuntu - http://www.ubuntu.com/
[6] Unicode - http://unicode.org
[7] Nepali Unicode Locale – http://www.mpp.org.np
[8] Roman to Unicode converter - http://www.unicodenepali.com/convert/

Personal:

[1] James L. Frankel, Mitshubishi Electrical Research lab, USA
[2] Shishir Basyal, University of Missourie, USA
[3] Walter Bender, One Laptop Per Child, USA
[4] Marco Pessenti Gritti, Redhat Inc. , Fedora Project, OLPC, USA
[5] Prakesh S. Adhikari, Yogesh Shrestha, Pawan K. Adhikari, Kiran Dhakal, IFCD, Nepal
[6] Jaya Prashad Lamsal, Curriculum Development Center, Nepal
[7] Dwarika Bhattarai, Nepal Engineering College, Nepal
[8] Shailesh Bdr. Pandey, Nepal Engineering College, Nepal
[9] Parash Pradhan, Subir Pradhananga, Madan Puraskar Pustakalaya, Nepal